NUS Assignment by Syed Salman Rabbani | Week 2

Customer Segmentation using K-Means Clustering

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import silhouette_samples
```

Loading File

```
df = pd.read_csv('customer_data.csv')
df.head()
df.info()
df.describe()
```

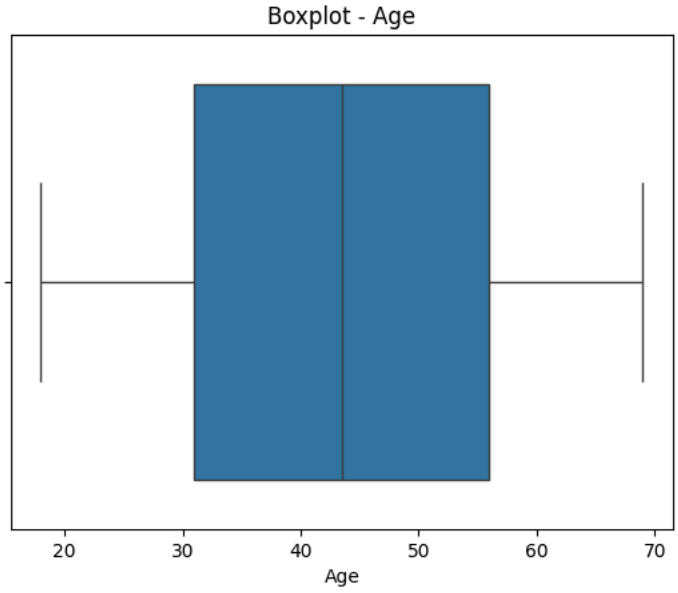
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Age                     200 non-null    int64
 2   Annual Income (k$)      200 non-null    int64
 3   Spending Score (1-100)  200 non-null    float64
 4   Purchase Frequency      200 non-null    float64
 5   Avg Purchase Value      200 non-null    float64
dtypes: float64(3), int64(3)
memory usage: 9.5 KB
```

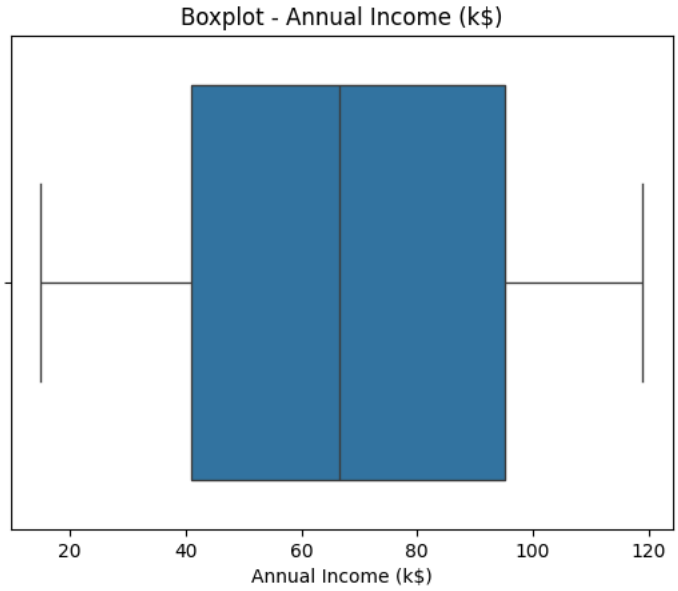
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	Purchase Frequency	Avg Purchase Value
count	200.000000	200.00000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	43.42500	67.145000	50.869302	5.335500	43.092021
std	57.879185	14.94191	31.249587	22.563855	2.687808	29.326249
min	1.000000	18.00000	15.000000	1.000000	0.600000	10.000000
25%	50.750000	31.00000	41.000000	32.366332	3.000000	17.651673
50%	100.500000	43.50000	66.500000	52.352457	5.450000	36.086311
75%	150.250000	56.00000	95.250000	67.224241	7.600000	59.641832
max	200.000000	69.00000	119.000000	100.000000	10.000000	137.621150

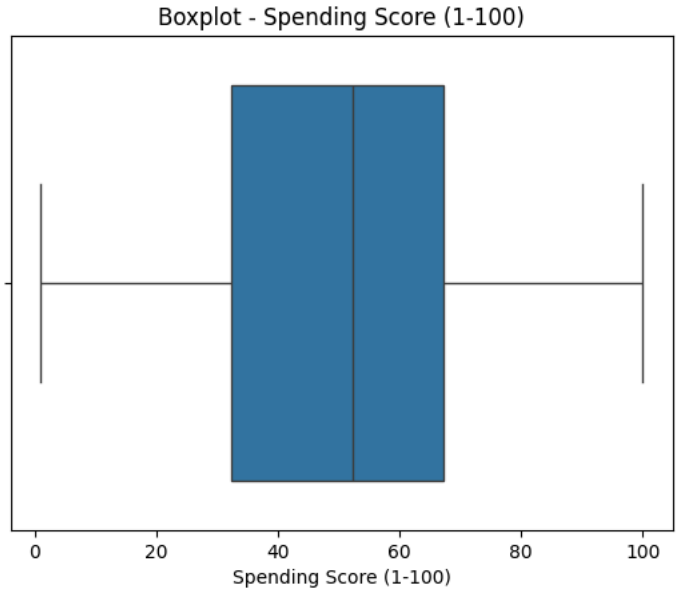
EDA

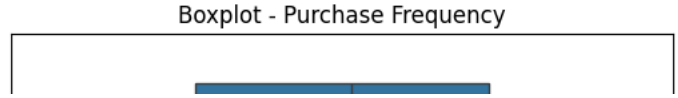
```
# Check for missing values
print(df.isnull().sum())

# Boxplots for outliers
features = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)',
            'Purchase Frequency', 'Avg Purchase Value']

for feature in features:
    sns.boxplot(x=df[feature])
    plt.title(f'Boxplot - {feature}')
    plt.show()
```

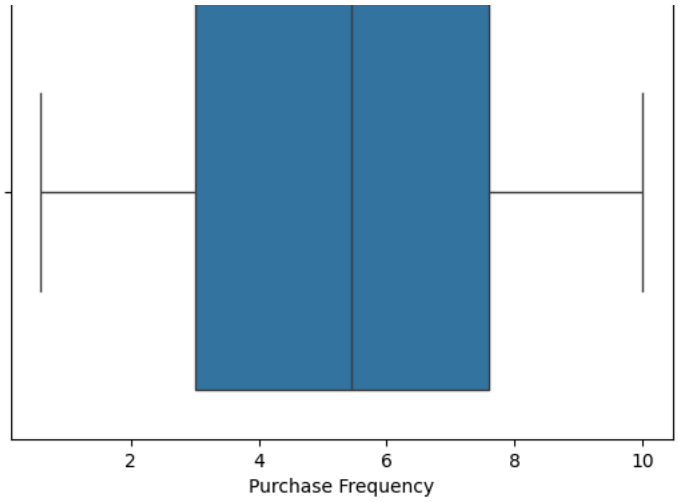
```
CustomerID                0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
Purchase Frequency        0
Avg Purchase Value        0
dtype: int64
```

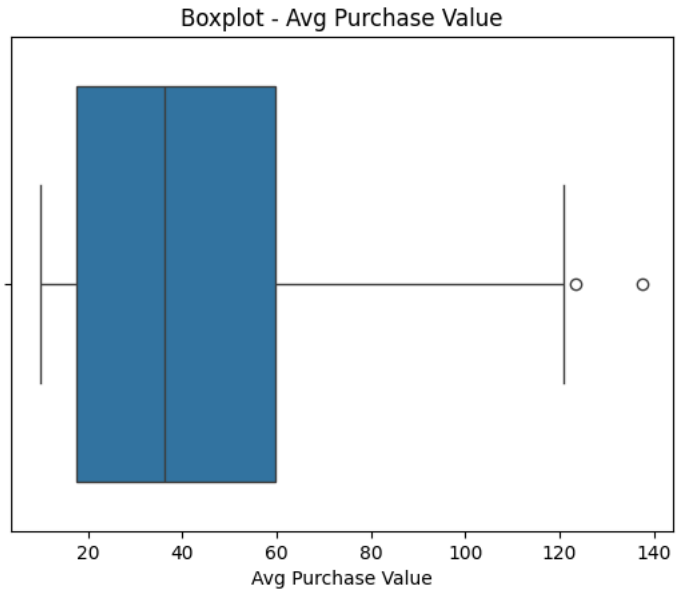






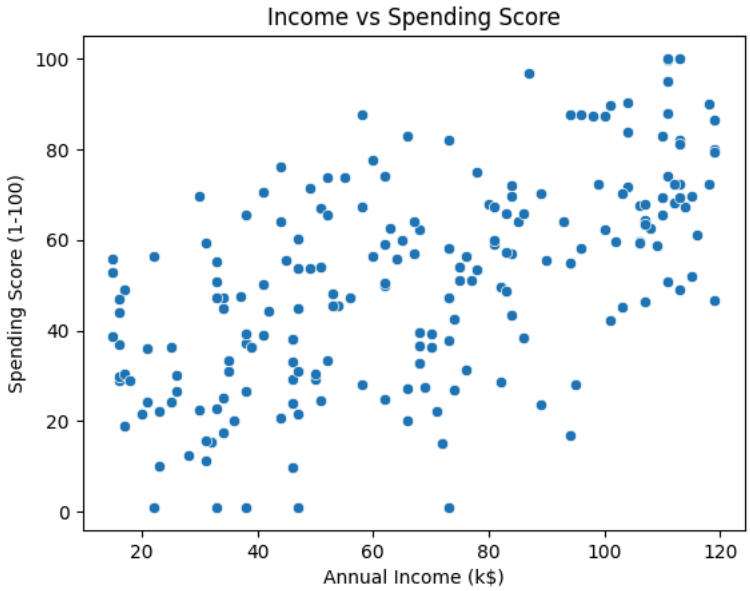


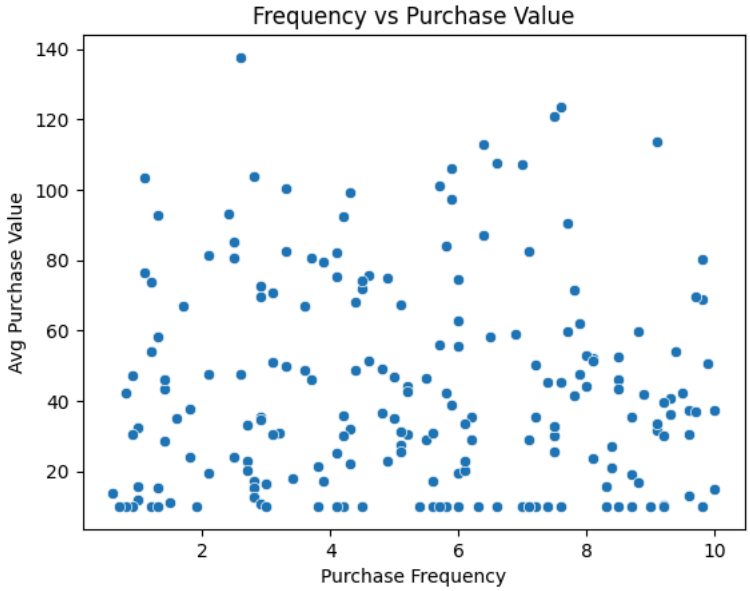

2
4
6
8
10
Purchase Frequency

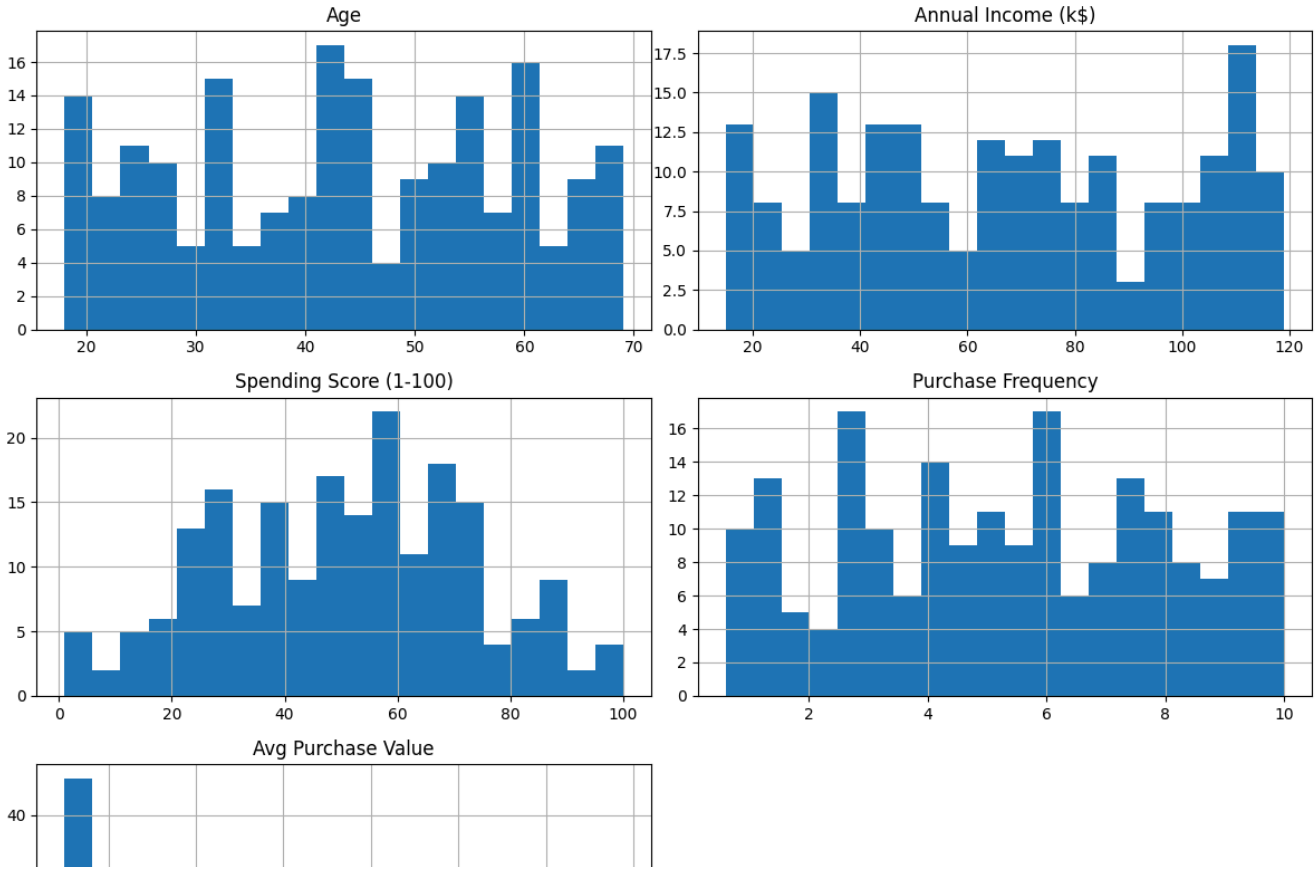

Boxplot - Avg Purchase Value
20
40
60
80
100
120
140
Avg Purchase Value

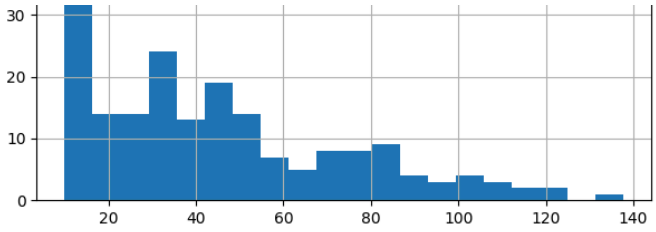
Visualization

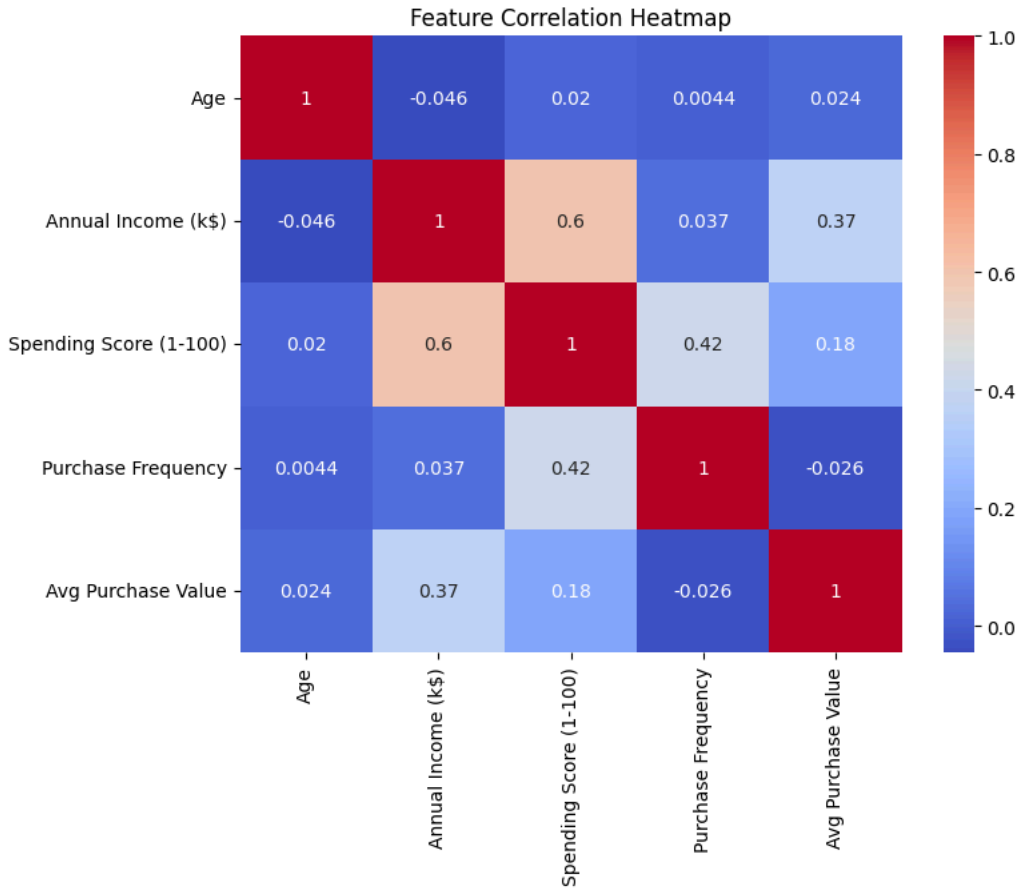
```
# Scatter plots
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)', data=df)
plt.title('Income vs Spending Score')
plt.show()

sns.scatterplot(x='Purchase Frequency', y='Avg Purchase Value', data=df)
plt.title('Frequency vs Purchase Value')
plt.show()

# Distribution plots
df[features].hist(bins=20, figsize=(12, 10))
plt.tight_layout()
plt.show()

# Correlation heatmap
plt.figure(figsize=(8,6))
sns.heatmap(df[features].corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()
```


Income vs Spending Score
Spending Score (1-100)
Annual Income (k$)


Frequency vs Purchase Value
Avg Purchase Value
Purchase Frequency


Age
Annual Income (k$)
Spending Score (1-100)
Purchase Frequency
Avg Purchase Value


30
20
10
0
20
40
60
80
100
120
140


Feature Correlation Heatmap
Age
Annual Income (k$)
Spending Score (1-100)
Purchase Frequency
Avg Purchase Value
1
-0.046
0.02
0.0044
0.024
-0.046
1
0.6
0.037
0.37
0.02
0.6
1
0.42
0.18
0.0044
0.037
0.42
1
-0.026
0.024
0.37
0.18
-0.026
1
1.0
0.8
0.6
0.4
0.2
0.0

Data Preprocessing

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[features])
```

To Find Optimal Number of Clusters

```
inertia = []
silhouette_scores = []
K_range = range(2, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X_scaled, kmeans.labels_))

plt.plot(K_range, inertia, marker='o')
plt.title('Elbow Method')
plt.xlabel('k')
plt.ylabel('Inertia')
plt.show()

plt.plot(K_range, silhouette_scores, marker='o')
plt.title('Silhouette Scores')
plt.xlabel('k')
plt.ylabel('Score')
plt.show()
```